Control Rules for Reactive System Games *

Matteo Slanina
Stanford University, Computer Science Department
Stanford, CA 94305–9045
E-mail: matteo@cs.stanford.edu

Abstract

This paper presents a deductive approach to the control problem for infinite-state reactive systems. It describes three proof rules, sound and relatively complete for formulas in the first two levels of the hierarchy of linear temporal logic—safety and response. The control conditions forming the premises of the rules are Π_2^0 first-order formulas. If a subroutine can prove their validity constructively, the extracted programs can be used to synthesize a winning strategy for the controller.

Introduction

Although the synthesis problem for concurrent systems interested researchers before the problem of their verification (Church 1962; Rabin 1969), it later succumbed to the wide spreading of applications of the latter. The main reason is the high computational complexity associated with all forms of finite-state reactive program synthesis: completeness for 2EXPTIME is a standard lower bound (Pnueli & Rosner 1989b; Kupferman & Vardi 1999; 2000).

Nevertheless, among the numerous variations of the reactive program synthesis problem (Anuchitanukul 1995; Kupferman & Vardi 1999), there is at least one area wherein solutions are practically needed and has been the subject of active research in recent years: control and the synthesis of controlling strategies.

Consider a system made of many concurrent processes running in parallel. Some processes have a program that is fixed, either by another designer, or because it is the most realistic model of some physical process. The control problem is to design, given appropriate restrictions, programs for the remaining modules, in such a way that the interaction of all components obey some given properties, specified by a formula in some formal language—typically temporal logic. The synthesized controller is essentially a winning strategy in an infinite game, and the control problem is intimately related to that of solving infinite games (Pnueli & Rosner 1989a; Zielonka 1998).

The well-known approaches to the control problem are algorithmic and can only be applied to finite-state systems (Anuchitanukul 1995; Kupferman *et al.* 2000). This paper uses deductive techniques for proving controllability of infinite-state reactive systems. I introduce a general model, which extends fair transition systems (FTS) (Manna & Pnueli 1991), and then present a series of deductive rules that allow us to reduce the question of controllability—with respect to certain classes of properties—to validity problems of the underlying assertion language. This mirrors similar techniques for verification of FTS's. Whereas for verification of FTS's the resulting verification conditions are quantifier-free, part of the control conditions resulting from our techniques are Π_2^0 formulas, i.e., of the form $\forall \vec{x}\, \exists \vec{y}\, \varphi(\vec{x}, \vec{y})$. Overall, thus, the methods outlined show how to reduce control of reactive systems to small functional synthesis problems. These can be solved with standard constructive theorem proving methods, or, in cases where the assertion language permits it, by specialized decision procedures.

If the assertional prover succeeds and returns synthesized functions that realize the conditions, each rule allows to synthesize, from these, a program for the controlling process of the reactive system.

I prove relative completeness of the proposed rules: if a system is controllable with respect to a property of a certain class, then this fact can be proved by suitable premises of the respective rule, valid with respect to the assertion language. The assertion language must be powerful enough to enable encoding of finite sequences and contain fixpoint operators. This is exactly the language necessary to prove completeness of the corresponding rules for verification. I also show how, in some special cases, relative completeness can be established for a less powerful language that does not contain

*This research was supported in part by NSF(ITR) grant CCR-01-21403, by NSF grant CCR-99-00984-001, by ARO grant DAAD19-01-1-0723, and by ARPA/AF contracts F33615-00-C-1693 and F33615-99-C-3014.

fixpoint operators.

The completeness proofs, and much of the intuition behind the rules, rely on results about games on finitely colored graphs (Zielonka 1998).

The rest of the paper is organized as follows: First, we introduce the computation model, the temporal logic used as specification language, and other relevant notation. Although all the notation used in the paper is shortly described, the section is very probably not self-contained, but a previous acquaintance with transition systems and the temporal logic LTL is necessary; pointers to the relevant literature are found in the references. In the next, major, section, the deductive rules are introduced, together with proofs of soundness and some very short examples. Some of the completeness proofs are given in the appendix, after a conclusion and a bibliography sections.

Computational Model and Specification

Game transition systems. Our computation model is an extended version of Manna and Pnueli's fair transition systems (Manna & Pnueli 1991), that we call *game transition systems* (GTS). It is fairly general, allowing to model both synchronous and asynchronous concurrency and different kinds of fair scheduling. For another, similar, model appearing in the literature see (Alur, Henzinger, & Kupferman 1997).

We assume an underlying *assertion language*, a typed first-order language with standard interpretation for the types. For example, the language of Peano Arithmetic, or a language with integers and arrays with their standard interpretations, would serve our purpose. The language can also have least and greatest fixpoint operators on positive formulas, and this will be instrumental in the proofs of relative completeness. Given an interpretation for the variables of the assertion language, or *state*, s, we write $s \models p$, or "s is a p-state", to say that the assertion formula φ is true in state s.

A game transition system consists of states and transitions. States are modeled as assignments to a predefined set of logical variables in an underlying assertion language. Transitions model concurrently executing processes. In this paper we consider systems with two players: Player 2's behavior will be fixed in advance, while we are free to synthesize behaviors for Player 1. The transitions are partitioned between the two players and there is a third, hidden, player, the Scheduler, which repeatedly chooses an enabled transition and allows the player that owns that transition to move to another state. We shall assume a worst-case scenario, that is, our goal will be to synthesize a strategy for Player 1 that realizes the goal independently of the choices of Player 2 *and* the Scheduler; in other words, we assume Player 2 and the Scheduler collaborate against Player 1. This is contrary to the commoner approach, where the controller is always a single process (transition). Allowing the controller to consist of more processes, whose scheduling is controlled by an adversary—although an adversary restricted by fairness—does not add to the complexity of the proof rules, while allowing to describe questions like, e.g., "What happens to a protocol when two of the participating processes misbehave, while all others adhere to the protocol? Can the two force the system to ...?"

Formally, a GTS G is a 6-tuple $\langle V, \Theta, \mathcal{T}_1, \mathcal{T}_2, \mathcal{J}, \mathcal{C}\rangle$, where:

- V is a finite set of typed variables in the underlying assertion language.
- Θ is an assertion on V characterizing the initial states of the GTS.
- $\mathcal{T}_1$ and $\mathcal{T}_2$ are finite sets of transitions, for Players 1 and 2, respectively. We shall denote $\mathcal{T}_1 \cup \mathcal{T}_2$ by $\mathcal{T}$. A transition $\tau \in \mathcal{T}$ characterizes a binary relation on states, represented by an assertion $\tau(V, V')$ on variables of V and primed copies thereof. For two states s and s', the meaning of $s, s' \models \tau(V, V')$ is that transition τ can lead from s to s' in a single step. For example, if τ is $|x - x'| = 1$, then τ is a transition that, if taken, either increases or decreases the value of x by 1.
- $\mathcal{J}$ and $\mathcal{C}$ are subsets of $\mathcal{T}$, denoting the fairness conditions. A transition $\tau \in \mathcal{J}$, called *just*, must eventually be scheduled if it is continually enabled from some point of an execution on. A transition $\tau \in \mathcal{C}$, called *compassionate*, must eventually be scheduled if it is enabled infinitely many times, even if not continually. Clearly, every compassionate transition is also just.

Example. The following GTS, FLOW, models a turn game between two players that alternatively change a real-valued parameter x. Player 1 can change the value by at most 2 in each step, Player 2 by at most 1. (Think of it as a random flow, between -1 and 1 unit per hour in value, going in or out of a reservoir, and Player 1 being able to correct the value by opening some valves once per hour.) The GTS is defined on an alphabet of two variables, x : real and $\mathit{turn} : \{1, 2\}$, with the initial condition being $\Theta : x = 0$, and the sets of transitions $\mathcal{T}_1 = \{\tau_1\}$, $\mathcal{T}_2 = \{\tau_2\}$, where

$$\tau_1 : \mathit{turn} = 1 \wedge \mathit{turn}' = 2 \wedge |x' - x| \leq 2,$$
$$\tau_2 : \mathit{turn} = 2 \wedge \mathit{turn}' = 1 \wedge |x' - x| \leq 1.$$

Fairness conditions are clearly irrelevant here, so we can take $\mathcal{J} = \mathcal{C} = \emptyset$.

A *state* is an assignment of values to the variables in V. We assume standard interpretations for the types,

e.g., integers, arrays, etc. The collection of all states is denoted by Σ. A transition τ is *enabled* on a state s if some state s' can be reached from s through τ. This property is expressed by the assertion $En(\tau)$, defined as $\exists V'.\tau(V, V')$. Clearly, a transition τ is enabled on s if and only if $s \models En(\tau)$. We shall sometimes write $Tak(\tau)$ (for "taken") to mean $\tau(V, V')$.

We always assume that GTS's are non-blocking, i.e., there is some enabled transition in every state. Most systems are non-blocking by nature and need no check, but the assumption is not restrictive in any case, since being non-blocking is a standard invariance property, and can thus be checked by standard verification methods that are assumed by this paper.

A *computation*, or *play*, of G is a sequence of states and transitions $\sigma : s_0 \xrightarrow{\tau_0} s_1 \xrightarrow{\tau_1} s_2 \xrightarrow{\tau_2} \cdots$ such that

- $s_0 \models \Theta$;
- $s_i, s_{i+1} \models \tau_i$ for all i (this stays, more explicitly, for $[V/s_i, V'/s_{i+1}] \models \tau_i(V, V')$);
- if $\tau \in \mathcal{J}$ and $s_i \models En(\tau)$ for all $i \geq n$, then $\tau_i = \tau$ for some $i \geq n$ (justice);
- if $\tau \in \mathcal{C}$ and $s_i \models En(\mathcal{T})$ for infinitely many $i \geq n$, then $\tau_i = \tau$ for some $i \geq n$ (compassion).

A *strategy*, or program, for Player 1 (and analogously for Player 2) is a partial function $f : \Sigma^* \times \mathcal{T}_1 \rightharpoonup \Sigma$ such that $f(\sigma, \tau)$ is defined whenever τ is enabled on the last state, s, of σ, and then $s, f(\sigma, \tau) \models \tau$; that is, the strategy returns one of the τ-successor states according to. A play σ is *compatible* with a strategy f if $s_{i+1} = f(s_0 s_1 \ldots s_i, \tau_i)$ whenever $\tau_i \in \mathcal{T}_1$. If a strategy is *memoryless*, i.e., it only depends on the last state of a computation segment, we shall write, with abuse of notation, $f : \Sigma \times \mathcal{T}_1 \rightharpoonup \Sigma$.

A GTS $G = \langle V, \Theta, \mathcal{T}_1, \mathcal{T}_2, \mathcal{J}, \mathcal{C} \rangle$ can be thought of as an FTS $\hat{G} = \langle V, \Theta, \mathcal{T}, \mathcal{J}, \mathcal{C} \rangle$, disregarding the player distinctions. This identification allows us to use the proof techniques for FTS's on $\hat{G}$. Any LTL formula holding of $\hat{G}$ clearly holds of any play of G. This is useful because we can prove LTL properties (especially invariants) and use them as auxiliary lemmas in proofs of control conditions.

Specification language. As a specification language we use LTL, the linear time temporal logic, plus some notational conventions from ATL (alternating temporal logic).

An LTL formula describes a property of an infinite sequence of states, and there are operators to talk about the future and the past of the sequence with respect to a given reference point. Formally (in the notation of (Manna & Pnueli 1995)):

- every assertion is an LTL formula;
- if φ and ψ are LTL formulas, so are $\neg\varphi$, $\varphi \wedge \psi$, $\bigcirc\varphi$ (i.e., φ at the next time step), $\ominus\varphi$ (i.e., φ at the previous time step), $\varphi \mathcal{U} \psi$ (i.e., φ until ψ), and $\varphi \mathcal{S} \psi$ (i.e., φ since ψ).

Given a sequence $\sigma : s_0, s_1, s_2, \ldots$ of states, truth of an LTL formula, φ, at position i in σ, denoted $(\sigma, i) \vDash \varphi$, is defined as:

- $(\sigma, i) \vDash p$ if $s_i \models p$, for a state formula p;
- $(\sigma, i) \vDash \neg\varphi$ if $(\sigma, i) \nvDash \varphi$;
- $(\sigma, i) \vDash \varphi \wedge \psi$ if $(\sigma, i) \vDash \varphi$ and $(\sigma, i) \vDash \psi$;
- $(\sigma, i) \vDash \bigcirc\varphi$ if $(\sigma, i+1) \vDash \varphi$;
- $(\sigma, i) \vDash \ominus\varphi$ if $i > 0$ and $(\sigma, i-1) \vDash \varphi$;
- $(\sigma, i) \vDash \varphi \mathcal{U} \psi$ if there is a $j \geq i$ such that $(\sigma, j) \vDash \psi$, and $(\sigma, k) \vDash \varphi$ for all $k \in \{i, i+1, \ldots, j-1\}$;
- $(\sigma, i) \vDash \varphi \mathcal{S} \psi$ if there is a $j \leq i$ such that $(\sigma, j) \vDash \psi$, and $(\sigma, k) \vDash \varphi$ for all $k \in \{j+1, j+2, \ldots, i\}$.

We write $\sigma \vDash \varphi$ for $(\sigma, 0) \vDash \varphi$. The abbreviations $\diamondsuit\varphi$ (eventually φ) and $\square\varphi$ (henceforth φ) stand for $\text{TRUE}\,\mathcal{U}\,\varphi$ and $\neg\diamondsuit\neg\varphi$, respectively. An LTL formula is a *future* (resp. *past*) *formula* if it does not contain any past (resp. future) operator: $\ominus, \mathcal{S}$ (resp. $\bigcirc, \mathcal{U}, \square, \diamondsuit$).

We say that Player 1 can *win* on the GTS G with winning condition φ, where φ is an LTL formula, if there is a strategy, f, for Player 1 such that $\sigma \vDash \varphi$ for all plays σ consistent with f. Following (Alur, Henzinger, & Kupferman 1997), we write $G \vDash \langle\langle 1 \rangle\rangle \varphi$ if this is the case. For example, $G \vDash \langle\langle 1 \rangle\rangle \diamondsuit (state = checkmate)$ denotes that Player 1 has a strategy to force the game to eventually reach a state where variable *state* equals value *checkmate*.

A more powerful logic to talk about games and strategies is ATL (Alur, Henzinger, & Kupferman 1997). ATL extends LTL by allowing arbitrary nesting of $\langle\langle A \rangle\rangle$ and LTL operators, where A is a set of players. The formula $\langle\langle A \rangle\rangle \varphi$ means "the players in A have a collaborative strategy to force φ, no matter what the remaining players do". Thus, our $\langle\langle 1 \rangle\rangle$ corresponds to $\langle\langle \{1\} \rangle\rangle$, and $\langle\langle \emptyset \rangle\rangle$ corresponds to the CTL* universal path quantifier A.

In this paper we shall only use the $\langle\langle 1 \rangle\rangle$ operator—and never in nested form—and the $\langle\langle \emptyset \rangle\rangle$ operator, which, for notational convenience, we shall generally omit: thus, if a part of the formula is not prefixed by $\langle\langle 1 \rangle\rangle$, it should be interpreted as evaluated in $\hat{G}$, or, equivalently, as being prefixed by $\langle\langle \emptyset \rangle\rangle$—i.e., the formula predicates over all computations. For example, a *strong response* formula is of the form $\square(p \rightarrow \langle\langle 1 \rangle\rangle \diamondsuit q)$, for p and q state formulas. In full ATL it would be written $\langle\langle \emptyset \rangle\rangle \square(p \rightarrow \langle\langle 1 \rangle\rangle \diamondsuit q)$, and it holds in a GTS G if from all reachable p-states Player 1 has a strategy to reach a q-state.

Premises of rules, if not starting by any strategy or temporal operator, should always be considered prefixed by $\langle\langle\emptyset\rangle\rangle\Box$.

Hoare triples & co. Some abbreviations will make our notation much more manageable.

For a state formula p, let p' be a copy of p where all variable names have been primed. For a past formula φ, we define φ' as

- $(\neg\varphi)' \equiv \neg(\varphi')$;
- $(\varphi \wedge \psi)' \equiv (\varphi') \wedge (\psi')$;
- $(\ominus\varphi)' \equiv \varphi$;
- $(\varphi \,\mathcal{S}\, \psi)' \equiv \psi' \vee (\varphi' \wedge \varphi \,\mathcal{S}\, \psi)$.

Intuitively, φ' denotes the formula φ evaluated at the next state instead of the current one.

A (universal) Hoare triple $\{\varphi\}\,\tau\,\{\psi\}$, for past formulas φ and ψ and transition τ, is defined as

$$\forall V.\, \varphi(V) \rightarrow \forall V'.\, \tau(V, V') \rightarrow \psi'(V').$$

It is well-known that if $G \vDash \Box\{\varphi\}\,\tau\,\{\psi\}$ (that is, in full ATL notation for the last time, $G \vDash \langle\langle\emptyset\rangle\rangle\Box\{\varphi\}\,\tau\,\{\psi\}$) and σ is a φ-computation prefix of G ending in a state s, then $\sigma \cdot s' \vDash \psi$ for every state s' reachable from s through τ.

In (Sipma 1999) and (Bjørner *et al.* 2001), the authors introduced a variant called *existential Hoare triples*, which we use with a slight modification. An existential Hoare triple $\{\varphi\}\,\tau\exists\,\{\psi\}$ is the formula

$$\forall V.\, \varphi(V) \wedge \mathit{En}(\tau)(V) \rightarrow \exists V'.\, \tau(V, V') \wedge \psi'(V').$$

It is easy to see that if $G \vDash \Box\{\varphi\}\,\tau\exists\,\{\psi\}$ and σ is a φ-computation-prefix of G ending in a state s, where τ is enabled, then there is a τ-successor s' of s such that $\sigma \cdot s' \vDash \psi$.

Finally, we take the *control Hoare triple* $\{\!\{\varphi\}\!\}\,\tau\,\{\!\{\psi\}\!\}$ to mean $\{\varphi\}\,\tau\,\{\psi\}$, if $\tau \in \mathcal{T}_2$, or $\{\varphi\}\,\tau\exists\,\{\psi\}$, if $\tau \in \mathcal{T}_1$ (remember our objective is to find a strategy for Player 1). From the previous two observations we conclude that if $G \vDash \Box\{\!\{\varphi\}\!\}\,\tau\,\{\!\{\psi\}\!\}$ then Player 1 has a (one-step) strategy, f, such that, from every φ-segment, if τ is taken and the computation is extended compatibly with f, the extended segment satisfies ψ at the last position.

The abbreviation $\{\!\{\varphi\}\!\}\,\mathcal{T}\,\{\!\{\psi\}\!\}$ means

$$\{\!\{\varphi\}\!\}\,\tau\,\{\!\{\psi\}\!\} \text{ for all } \tau \in \mathcal{T},$$

and analogously for universal and existential Hoare triples.

Rules for Control

Any LTL property is equivalent to a canonical form described in (Manna & Pnueli 1989; 1991). LTL properties can be classified into the following hierarchy:

- safety: $\Box p$, for p a past formula;
- response: $\Box(p \rightarrow \Diamond q)$, for p and q past formulas;
- reactivity: $\bigwedge_{i=1}^{n}(\Box\Diamond p_i \vee \Diamond\Box q_i)$, where p_i, q_i, for $i \in \{1, \ldots, n\}$, are past formulas.

In (Manna & Pnueli 1989), the authors give proof rules for the verification of each class of properties over FTS's. We want to extend their results by showing how to prove controllability of a GTS with goal $\langle\langle 1\rangle\rangle\varphi$—where φ is a property in one of the above classes—or some variations thereof more specific to game properties.

To save space and ease exposition, we assume in most proofs that p and q are state formulas. When not otherwise noted, the results hold for the general case of them being past formulas, with minor changes in the proofs. An effective alternative to using past formulas is to augment the program with *history variables* (Manna & Pnueli 1989; 1995) and thus transform the problem into an equivalent one with only future formulas. This can be done in control in the same way as it is done in verification.

Safety. The well-known method to prove that an FTS S satisfies some invariance property $\Box p$ is to find a strengthening of p, say φ, that is preserved by all transitions of S. The resulting rule, which can be thought of as a form of induction, can be written as

$$\textsc{inv}:\quad \frac{\Theta \rightarrow \varphi \qquad \{\varphi\}\,\mathcal{T}\,\{\varphi\} \qquad \varphi \rightarrow p}{S \vDash \Box p}$$

where the premises are, as usually, to be proved with respect to S. This rule is sound and complete for proving all invariance properties of the system, relatively to validity in the underlying assertion language.

Analogously, to prove that $G \vDash \langle\langle 1\rangle\rangle\Box p$, it suffices to find an assertion φ, stronger than p, that is preserved by any move of Player 2 and *can* be preserved by Player 1 when he is scheduled to move. This idea gives us the following rule:

$$\textsc{inv-control}:\quad \frac{\Theta \rightarrow \varphi \qquad \{\!\{\varphi\}\!\}\,\mathcal{T}\,\{\!\{\varphi\}\!\} \qquad \varphi \rightarrow p}{G \vDash \langle\langle 1\rangle\rangle\Box p}$$

Example. Consider the GTS, FLOW, of our first example, and suppose we want to prove that FLOW $\vDash \langle\langle 1\rangle\rangle\Box(|x| \leq 1)$. Then we can choose the strengthening φ to be

$$\varphi : (\mathit{turn} = 2 \wedge |x| \leq 1) \vee (\mathit{turn} = 1 \wedge x = 0).$$

Theorem 1 (Soundness of INV-CONTROL). *If the premises of* INV-CONTROL *are G-state valid, then the conclusion holds.*

Proof. Assume the premises hold. Let f be the following (memoryless) strategy for Player 1:

$$f(s,\tau) = \begin{array}{l}\text{if } s \text{ is a reachable } \varphi\text{-state,}\\ \text{then some } s' \text{ such that } s' \models \varphi,\\ \text{else an arbitrary state.}\end{array}$$

The function f is well-defined because of the G-validity of the verification conditions. Any play compatible with f consists of φ-states only, and therefore, but the third premise of INV-CONTROL, of p-states only, which proves $\langle\!\langle 1\rangle\!\rangle \Box p$. □

The following theorem is proved in the appendix.

Theorem 2 (Relative completeness of INV-CONTROL). INV-CONTROL *is complete relatively to assertional validities, i.e., if $G \models \langle\!\langle 1\rangle\!\rangle \Box p$, then there is an assertion φ such that the premises of rule* INV-CONTROL *can be proved from assertional validities.*

If p is a past formula, the results still hold, by allowing φ to also be a past formula.

Response. An LTL response formula is of the form $\Box(p \rightarrow \Diamond q)$, with p and q past formulas in general, but for now we assume they are state formulas. To this class belong, for example, formulas expressing reachability properties, termination of programs, response to single stimuli.

Contrary to the case of invariance, there are two possible interpretations of the response class into control problems. The first one, which we shall call *weak response*, can be expressed by the formula $\langle\!\langle 1\rangle\!\rangle \Box(p \rightarrow \Diamond q)$. The second one, *strong response*, is expressed by $\Box(p \rightarrow \langle\!\langle 1\rangle\!\rangle \Diamond q)$. Player 1 can win a strong response game if he has a strategy to guide the game into a q-state from *any* reachable p-state. To win a weak response game, on the other hand, less is required from him: he can, for example, use a strategy that keeps the game into $(\neg p)$-states forever, if he has one.

We shall now deal with these two kinds of games separately. As for the case of verification, if controllability depends on compassion, some of the control conditions will be simpler temporal control problems—precisely, strong response problems.

Two important issues arise that were not present in invariance problems: the first, dependence of the properties on fairness conditions, was already present in verification. The other, the existence or lack of memoryless strategies, is peculiar to control.

Strong response. Suppose we are given a GTS $G = \langle V, \Theta, \mathcal{T}_1, \mathcal{T}_2, \mathcal{J}, \mathcal{C}\rangle$ and a strong response formula $\varphi : \Box(p \rightarrow \langle\!\langle 1\rangle\!\rangle \Diamond q)$. Extending the F-RESP rule of (Manna & Pnueli 1989), we introduce the rule S-RESP-CONTROL.

Let $\tau_1, \ldots, \tau_m$ be a list of the just and compassionate transitions of G. To apply the rule, we need to find auxiliary properties $\varphi_1, \ldots, \varphi_m$—one φ_i corresponding to one τ_i—and a ranking function $\delta : \Sigma \rightarrow A$, mapping states into the domain of a well-founded order $\langle A, \preceq\rangle$. Let φ be an abbreviation for $\varphi_1 \vee \cdots \vee \varphi_m$. (As a special case, if $\mathcal{J} \cup \mathcal{C} = \emptyset$, we are allowed to choose φ as an auxiliary property.) Then we have the rule

S-RESP-CONTROL:

$p \rightarrow q \vee \varphi$

For all $\tau_i \in \mathcal{J} \cup \mathcal{C}$:

$\{\!\{\varphi_i \wedge \delta = a\}\!\}\, \mathcal{T}\, \{\!\{q \vee (\varphi \wedge \delta \prec a) \vee (\varphi_i \wedge \delta \preceq a)\}\!\}$

$\{\!\{\varphi_i \wedge \delta = a\}\!\}\, \tau_i\, \{\!\{q \vee (\varphi \wedge \delta \prec a)\}\!\}$

if $\tau_i \in \mathcal{J}$,

$\varphi_i \rightarrow q \vee En(\tau_i)$

if $\tau_i \in \mathcal{C}$,

$$\frac{G^i \models \Box(\varphi_i \wedge \delta = a \rightarrow \langle\!\langle 1\rangle\!\rangle \Diamond(q \vee (\varphi \wedge \delta \prec a) \vee (En(\tau_i) \wedge \varphi_i \wedge \delta \preceq a)))}{G \models \Box(p \rightarrow \langle\!\langle 1\rangle\!\rangle \Diamond q)}$$

It is sufficient that the premises (all but the last) hold in all reachable states of G, i.e., that they are G-state valid. In the last premise, G^i is identical to G, apart from the fact that transition τ_i is replaced by $\tilde{\tau}_i$, where

$$\tilde{\tau}_i(V, V') \equiv \tau_i(V, V') \wedge \varphi_i(V) \wedge \delta(V) \preceq a,$$

and τ_i (or, $\tilde{\tau}_i$) is taken out of $\mathcal{C}$ (made an unfair transition). Thus the last premise has to be proved with respect to a GTS with one compassionate transition less than the original one, and there is no circularity in the rule.

Intuitively, δ is a ranking function that Player 1 can guarantee never to increase until a q-state is reached. Moreover, for any plateau in the computation, i.e., segment where δ stays constant, some φ_i must hold throughout the time the computation stays on the plateau. This ensures that, by fairness, τ_i will eventually be taken, making δ finally decrease.

Example. Consider once again the GTS FLOW and suppose, this time, we want to prove FLOW $\models \Box\langle\!\langle 1\rangle\!\rangle \Diamond(x = 0)$, i.e., $\Box(true \rightarrow \langle\!\langle 1\rangle\!\rangle \Diamond(x = 0))$. Since $\mathcal{J} = \mathcal{C} = \emptyset$, we are allowed to choose an invariant φ without having distinguished $\varphi_1, \ldots, \varphi_m$. We then must choose $\langle A, \preceq\rangle$ and a ranking function δ. The control conditions become

$$\Box(true \rightarrow \varphi)$$
$$\{\varphi \wedge \delta = a\}\, \tau_1 \exists\, \{x = 0 \vee (\varphi \wedge \delta \prec a)\}$$
$$\{\varphi \wedge \delta = a\}\, \tau_2\, \{x = 0 \vee (\varphi \wedge \delta \prec a)\}$$

It is easy to check that, choosing φ to be *true*, the well-founded domain to be $\langle A, \preceq \rangle = \langle \mathbb{N}, \leq \rangle \times \langle \{1,2\}, \leq \rangle$, with the product ordered lexicographically, and δ to be $\langle \lfloor |x| \rfloor + turn, turn \rangle$, makes all three control conditions assertionally valid.

Theorem 3 (Soundness of S-RESP-CONTROL). *If the premises of* S-RESP-CONTROL *are valid with respect to G, then the conclusion is valid with respect to G.*

Proof. Assume the premises hold with respect to G. Then we have, from the last premise, one substrategy for each pair $\langle i, a \rangle$, for $i \in \{1, \ldots, |\mathcal{C}|\}$ and $a \in A$. We define a winning strategy, f, for Player 1. When some particular conditions are satisfied, the player will stop playing f and play substrategy $\langle i, a \rangle$ instead, until some other condition is reached; at that point, Player 1 resumes playing f:

$f(s, \tau) =$
if s is not a φ-state, then irrelevant; otherwise,
if there is a q-state, s', reachable from s via τ,
  then s'; otherwise,
if there is a φ-state, s'', reachable from s via τ and
  such that $\delta(s'') \prec \delta(s)$, then s''; otherwise,
if $\tau_i \in \mathcal{J}$, then a φ_i-state, s''', reachable from s via τ
  and such that $\delta(s''') \preceq \delta(s)$;
if $\tau_i \in \mathcal{C}$, play substrategy $\langle i, \delta(s) \rangle$ until reaching a
  $q \vee (\varphi \wedge \delta \prec a) \vee (En(\tau_i) \wedge \varphi_i \wedge \delta \preceq a)$-state,
  then start playing f again.

The function f is well-defined because of the G-validity of the verification conditions.

We consider a play, $\sigma : s_0 \stackrel{\tau_0}{\rightarrow} s_1 \stackrel{\tau_1}{\rightarrow} \cdots$, compatible with f and its substrategies, and a position i such that $s_i \models p$ but the computation suffix $\sigma^i : s_i \stackrel{\tau_i}{\rightarrow} s_{i+1} \stackrel{\tau_{i+1}}{\rightarrow} \cdots$ consists of $(\neg q)$-states only, and try to derive a contradiction.

Since $\preceq$ is well-founded, there must be a minimal value, a, that δ assumes on the states of σ^i that are not internal to the substrategies. Let $j \geq i$ be a position such that $\delta(s_j) = a$. It must be the case that $s_j \models \varphi_k$, for some k such that $\tau_k \in \mathcal{J} \cup \mathcal{C}$. From the control conditions and the fact that a is minimal it follows that τ_k is never taken in σ^j, apart, possibly, during the execution of a substrategy. In case $\tau_k \in \mathcal{C}$, σ is a computation of G^k as well. The control conditions ensure that τ_k is enabled continually, if $\tau_k \in \mathcal{J}$, or infinitely many times, if $\tau_k \in \mathcal{C}$, on σ^j. In both cases, since σ is a fair computation, τ_k must eventually be taken. For $\tau_k \in \mathcal{C}$, in particular, τ_k must be taken in a state where $\varphi_k \wedge \delta \preceq a$, by definition of G^k. In any case, when τ_k is taken, δ decreases, contradicting the minimality of a. □

The following theorem is proved in the appendix.

Theorem 4 (Relative completeness of S-RESP-CONTROL). *Rule* S-RESP-CONTROL *is complete relatively to assertional validities, i.e., if $G \models \Box(p \rightarrow \langle\langle 1 \rangle\rangle \diamondsuit q)$, then there are assertions $\varphi_1, \ldots, \varphi_m$, a well-founded order $\langle A, \preceq \rangle$, and a ranking function $\delta : \Sigma \rightarrow A$—expressible in the assertion language—such that the premises of rule* INV-CONTROL *can be proved from state validities.*

As for invariance, the rule can be used with p, q, φ_i, ψ past formulas and δ depending on the past, as well.

Weak response. The rule to prove weak response properties is similar to the one for strong response. The difference is that we have an additional auxiliary formula ψ. Intuitively, ψ denotes computation segments where all occurrences of p in the past have already been followed by a q. Player 1 maneuvers in φ-segments, trying to decrease δ, as before, but, when all past requests (p) have been granted (q), he is also allowed to wander into ψ-segments.

W-RESP-CONTROL:
$\Theta \rightarrow \varphi \vee \psi$
$\{\!\{\psi\}\!\}\ \mathcal{T}\ \{\!\{\varphi \vee \psi\}\!\}$
$p \wedge \psi \rightarrow q \vee \varphi$,
and, for all $\tau_i \in \mathcal{J} \cup \mathcal{C}$,
$\{\!\{\varphi_i \wedge \delta = a\}\!\}\ \mathcal{T}\ \{\!\{(q \wedge (\varphi \vee \psi)) \vee (\varphi \wedge \delta \prec a) \vee (\varphi_i \wedge \delta \preceq a)\}\!\}$
$\{\!\{\varphi_i \wedge \delta = a\}\!\}\ \tau_i\ \{\!\{(q \wedge (\varphi \vee \psi)) \vee (\varphi \wedge \delta \prec a)\}\!\}$
If $\tau_i \in \mathcal{J}$,
$\varphi_i \rightarrow q \vee En(\tau_i)$
If $\tau_i \in \mathcal{C}$,
$G^i \models \Box(\varphi_i \wedge \delta = a \rightarrow \langle\langle 1 \rangle\rangle \diamondsuit((q \wedge (\varphi \vee \psi)) \vee (\varphi \wedge \delta \prec a) \vee (En(\tau_i) \wedge \varphi_i \wedge \delta \preceq a)))$
$\overline{G \models \langle\langle 1 \rangle\rangle \Box(p \rightarrow \diamondsuit q)}$

G^i is defined as in S-RESP-CONTROL.

Rule W-RESP-CONTROL is sound and complete for proving controllability with respect to weak response properties. The proofs are similar to those for strong response, but they are omitted due to space limitations. A caveat is that, even if p and q are state formulas, we may have to let ψ and φ be past formulas for completeness to hold.

Conclusions and Related Work

This paper shows how it is possible, through deductive rules, to reduce some important classes of control problems for reactive systems to problems of validity in an underlying assertion language. The salient point of the approach is its capacity to deal with infinite-state systems; another way to deal with infinitely many states in control, based on abstraction, is reported in (Henzinger *et al.* 2000).

Each of the rules presented can be recast in terms of verification (control) diagrams, as in (Manna & Pnueli 1994). This alternative formulation, which is still sound and complete, usually allows for a more intuitive, visual presentation of the structure of the proof. Because of space limitations, I defer a detailed exposition of control diagrams to an extended version of the paper.

Extensions to the present research will involve extending the class of properties to higher classes of the LTL hierarchy, i.e., reactivity properties, and possibly to more complex ATL properties. Diagrams are expected to be instrumental for the widest generalization. A general technique for the verification of LTL properties was given in (Manna *et al.* 1998): a generalized version of verification diagrams is there proved sound and relatively complete for the proof of any LTL property valid over an FTS. A similar result, for control of LTL or ATL properties over GTS's, can be expected as a possible final goal in this line of research. The final value of the proof method will, of course, rest on its applicability to serious examples and case studies.

Last, but not least, the proof rules depend heavily on being able to prove the control conditions, which are small functional synthesis problems. They will greatly benefit from advances in constructive theorem proving and decision procedures.

Acknowledgments

I am grateful to Bernd Finkbeiner, Tom Henzinger, Zohar Manna, César Sánchez, Sriram Sankaranarayanan, and Henny Sipma for helpful comments on an early draft of this paper.

References

Alur, R.; Henzinger, T. A.; and Kupferman, O. 1997. Alternating-time temporal logic. In *Proc. 38th IEEE Symp. Found. Comp. Sci.*, 100–109. An extended version appeared in *Compositionality—The Significant Difference*, LNCS 1536, pp. 23–60, Springer, 1999.

Anuchitanukul, A. 1995. *Synthesis of Reactive Programs*. Ph.D. Dissertation, Computer Science Department, Stanford University.

Bjørner, N. S.; Manna, Z.; Sipma, H. B.; and Uribe, T. E. 2001. Deductive verification of real-time systems using STeP. *Theoretical Computer Science* 253:27–60.

Church, A. 1962. Logic, arithmetic and automata. In *Proceedings of the International Congress of Mathematicians*, 23–35. Djursholm, Sweden: Institut Mittag-Leffler.

Henzinger, T. A.; Majumdar, R.; Mang, F.; and Raskin, J.-F. 2000. Abstract interpretation of game properties. In *Proc. 7th Intern. Static Analysis Symp. (SAS)*, volume 1824 of *LNCS*, 220–239. Springer.

Kupferman, O., and Vardi, M. Y. 1999. Church's problem revised. *The Bulletin of Symbolic Logic* 5(2):245–263.

Kupferman, O., and Vardi, M. Y. 2000. μ-calculus synthesis. In *Proc. 25th International Symposium on Mathematical Foundations of Computer Science*, volume 1893 of *LNCS*, 497–507. Springer.

Kupferman, O.; Madhusudan, P.; Thiagarajan, P.; and Vardi, M. 2000. Open systems in reactive environments: Control and synthesis. In *Proc. 11th Int. Conf. on Concurrency Theory*, volume 1877 of *LNCS*, 92–107. Springer.

Manna, Z., and Pnueli, A. 1989. Completing the temporal picture. In Ausiello, G.; Dezani-Ciancaglini, M.; and Ronchi Della Rocca, S., eds., *Proc. 16th Intl. Colloq. Aut. Lang. Prog.*, volume 372 of *LNCS*, 534–558. Springer. Also in Theoretical Computer Science.

Manna, Z., and Pnueli, A. 1991. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. New York: Springer.

Manna, Z., and Pnueli, A. 1994. Temporal verification diagrams. In Hagiya, M., and Mitchell, J. C., eds., *Proc. International Symposium on Theoretical Aspects of Computer Software*, volume 789 of *LNCS*, 726–765. Springer.

Manna, Z., and Pnueli, A. 1995. *Temporal Verification of Reactive Systems: Safety*. New York: Springer.

Manna, Z.; Browne, A.; Sipma, H. B.; and Uribe, T. E. 1998. Visual abstractions for temporal verification. In Haeberer, A., ed., *Algebraic Methodology and Software Technology (AMAST'98)*, volume 1548 of *LNCS*, 28–41. Springer.

Pnueli, A., and Rosner, R. 1989a. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. Princ. of Prog. Lang.*, 179–190.

Pnueli, A., and Rosner, R. 1989b. On the synthesis of an asynchronous reactive module. In *Proc. 16th Intl. Colloq. Aut. Lang. Prog.*, volume 372 of *LNCS*, 652–671. Springer.

Rabin, M. O. 1969. Decidability of second order theories and automata on infinite trees. *Trans. Amer. Math. Soc.* 141:1–35.

Sipma, H. B. 1999. *Diagram-Based Verification of Discrete, Real-Time and Hybrid Systems*. Ph.D. Dissertation, Computer Science Department, Stanford University.

Zielonka, W. 1998. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science* 200:135–183.

Appendix: Relative Completeness

Games on finitely colored graphs. All our completeness theorems rest on results on finite memory determinacy of GTS games. These results can be proved by translating the control problems into games on finitely colored graphs. We now review the basic notation and results that we shall use, following (Zielonka 1998).

An *arena* (or *game graph*) is a (possibly infinite) bipartite graph whose nodes are colored with colors from a finite set. Formally, an arena is a tuple $\langle V_1, V_2, E, C, c\rangle$, where $E \subseteq (V_1 \times V_2) \cup (V_2 \times V_1)$, for all $v \in V_1$ there is a $v' \in V_2$ such that $\langle v, v'\rangle \in E$, and similarly for vertices in V_2. We write V for $V_1 \cup V_2$. The set C is a finite nonempty set of colors and $c : V \to C$ is a coloring function. Zielonka deals with partial coloring functions, too, but we shall only need totally colored graphs in this paper.

A game is played by players alternatively moving a token from a vertex to an adjacent vertex—Player i moves at vertices in V_i—thus generating an infinite sequence of vertices. The winner is determined solely on the base of the set *inf* of colors that appear infinitely often in this infinite sequence. A Muller winning condition is a partition $\langle \mathcal{F}_1, \mathcal{F}_2\rangle$ of 2^C. Player i wins if, and only if, $inf \in \mathcal{F}_i$.

The main result of (Zielonka 1998) is that for every such game there is a partition $\langle W_1, W_2\rangle$ of the state space such that Player i has a finite memory strategy from each vertex in W_i. Moreover, if $\mathcal{F}_i$ can be expressed in a special form called *extended Rabin* condition, then Player i has a memoryless winning strategy from every vertex in W_i.

We can translate a GTS into a colored arena. We define V to be $\Sigma \cup (\Sigma \times \mathcal{T})$. The two players of the game graph represent Player 1 of the GTS, on one side, and Players 2 and Scheduler, on the other. We call the second player simply Player 2 when the context is clear enough. We define $V_1 = \Sigma \times \mathcal{T}_1$ and $V_2 = \Sigma \cup (\Sigma \times \mathcal{T}_2)$. The edges are $s \to \langle s, \tau\rangle$ for all τ such that $s \models En(\tau)$, and $\langle s, \tau\rangle \to s'$ for all s' such that $\tau(s, s')$. This arena, which is not bipartite, can be easily made such by adding dummy nodes, to be colored with the same color of their predecessors.

This basic game graph then undergoes further modification and coloring depending on the control property to check. The set of colors, C, will in general be the support of the Boolean algebra generated by some finite set of assertions, taken among the assertions appearing in the property to be checked, plus, possibly, assertions of the form $En(\tau)$ or $Tak(\tau)$, for $\tau \in \mathcal{J} \cup \mathcal{C}$. The coloring function is then the obvious one. The assertion $Tak(\tau)$ is taken to be true only in the vertices $\langle s, \tau\rangle$, for some $s \in \Sigma$. For example, if the chosen assertions are "$x \geq 0$", "$En(\tau)$", and "$Tak(\tau)$", where $En(\tau)$ is $x \equiv 0 \pmod 2$, then

$$c(x : 4) = \text{“}x \geq 0\text{”} \wedge \text{“}En(\tau)\text{”} \wedge \neg \text{“}Tak(\tau)\text{”},$$
$$c(\langle x : 4, \tau\rangle) = \text{“}x \geq 0\text{”} \wedge \text{“}En(\tau)\text{”} \wedge \text{“}Tak(\tau)\text{”}.$$

It is important to notice that the constructions of W_1, W_2, and winning strategies in (Zielonka 1998) are effectively representable in the assertion language. Moreover, the game graph can be easily expressed from the representation of a GTS. Thus, if we are able to capture the control problem by an appropriate coloring of the graph, Zielonka's theorems give us effective presentations of winning sets and winning strategies, which we can use in completeness proofs.

Invariance.

Theorem 2. *Rule* INV-CONTROL *is complete relatively to assertional validities, i.e., if* $G \models \langle\!\langle 1\rangle\!\rangle \Box p$*, then there is an assertion* φ *such that the premises of rule* INV-CONTROL *can be proved from state* (*not relative to* G) *validities.*

Proof. Remember we assumed the assertion language is powerful enough to encode finite sequences and have least and greatest fixpoint operators. We can easily modify the game graph construction sketched above (color the graph with two colors, p and $\neg p$, cut all successors of $(\neg p)$-vertices and replace them with self-loops) to capture the invariance control problem. It follows that the states of G can be partitioned into two sets, W_1 and W_2, such that Player 1 has a memoryless strategy, f, defined on every state of W_1, that allows him to maintain p forever, whilst Players 2 and S have a cooperative memoryless strategy, g, defined on each state in W_2, that allows them to reach a $\neg p$ state no matter what Player 1 does. Moreover, W_1 can be expressed in the assertion language, simply as the greatest fixpoint

$$W_1(V) = \nu X.p \wedge \bigwedge_{\tau \in \mathcal{T}_1} \big(En(\tau) \to \exists V'.\tau \wedge X(V')\big) \wedge \bigwedge_{\tau \in \mathcal{T}_2} \big(\forall V'.\,\tau \to X(V')\big)$$

It is easy to check that, taking φ to be W_1, the premises of the rule are valid. □

In the later completeness proofs I shall not give the exact expressions of the winning sets and strategies any more, but simply argue that the proofs of the theorems about existence of memoryless/finite memory winning strategies only use fixpoint constructions, weakest preconditions and other operators representable in the language.

The assertion language needed in the previous completeness proof is complex. Completeness for the analogous rule for verification, as shown in (Manna & Pnueli 1989), does not require fixpoint operators. On the other hand, fixpoints are needed for completeness of every other class of rules already in verification. As we shall see, we do not need to extend the language farther for other classes of control problems.

In some special cases, we can prove completeness of INV-CONTROL with respect to a language without fixpoints.

Theorem 5. *Let G be a* GTS *such that every $\tau \in \mathcal{T}_2$ is finitely branching, i.e., the set $\{s' \in \Sigma \mid \tau(s,s')\}$ is finite for all $s \in \Sigma$. Then* INV-CONTROL *is complete with respect to assertional validity in the language with finite sequencing, but no fixpoints.*

Proof sketch. The proof rests on the fact that if Player 2 has only finitely many moves from each position, then Player 1 can win the invariance game on G if and only if he can win the same game on every finite subgame of G, for a suitable definition of subgame encodable as a finite sequence. □

Response.

Theorem 4. *Rule* S-RESP-CONTROL *is complete relatively to assertional validities, i.e., if $G \vDash \Box(p \to \langle\langle 1\rangle\rangle \Diamond q)$, then there are assertions $\varphi_1, \ldots, \varphi_m$, a well-founded order $\langle A, \preceq\rangle$, and a ranking function $\delta : \Sigma \to A$—expressible in the assertion language—such that the premises of rule* S-RESP-CONTROL *can be proved from state validities.*

Proof. Assume that $G \vDash \Box(p \to \langle\langle 1\rangle\rangle \Diamond q)$.

We first want to find an expression *win* that characterizes the states in which $\langle\langle 1\rangle\rangle \Diamond q$ holds, i.e., from which Player 1 has a strategy that allows him to force the game to reach a q-state, and a list of expressions on V and V', one for each $\tau \in \mathcal{T}_1$, that characterize a memoryless strategy that is winning for the game $\langle\langle 1\rangle\rangle \Diamond q$ on each state in W_1.

To this end, we consider a colored arena for the game. We take the basic arena described in the first part of this section and cut off all successors of q-states, substituting them with self-loops. This will ensure that q-states are winning and we do not look beyond them. As a set of colors we choose the support of the Boolean algebra over the set of generators $\{q\} \cup \{En(\tau), Tak(\tau) \mid \tau \in \mathcal{J} \cup \mathcal{C}\}$. The coloring function is the one described in the basic construction. With this set of colors we can express the winning condition

$$\begin{aligned} &\bigvee_{\tau \in \mathcal{J}} inf \cap \Big(Tak(\tau) \cup \overline{En(\tau)} \Big) = \emptyset \\ &\vee \bigvee_{\tau \in \mathcal{C}} \big(inf \cap Tak(\tau) = \emptyset \big) \wedge \big(inf \cap En(\tau) \neq \emptyset \big) \\ &\vee inf \cap \overline{q} = \emptyset, \end{aligned} \tag{1}$$

which characterizes the ability of Player 1 to achieve $\langle\langle 1\rangle\rangle \Diamond q$ in all fair computations. Notice that winning condition (1) is in Rabin form. From results in (Zielonka 1998) we know, then, that Player 1 has a memoryless winning strategy from a set of states W_1, while Players 2 and Scheduler win from $W_2 = V \setminus W_1$ (they do not necessarily have a memoryless winning strategy, only a finite memory one). Moreover, W_1 and the memoryless winning strategy can be expressed as fixpoint formulas in the assertion language. Let *win* be the formula characterizing W_1. By hypothesis, all G-reachable p-states satisfy *win*.

The next step is to construct an FTS, $\tilde{G}$, a restriction of G where Player 1 only plays according to the aforementioned winning strategy; we shall then apply to G the completeness result of Pnueli (Manna & Pnueli 1989) for the verification of LTL response properties.

We build $\tilde{G}$ by conjoining the transition relation for each $\tau \in \mathcal{T}_1$ with the expression for the memoryless winning strategy, and making all G-reachable states initial in $\tilde{G}$ (reachability is expressible in the assertion language, see (Manna & Pnueli 1989; 1995) for details). The transitions for Player 2 and the fairness conditions do not change. In this way all transitions for Player 1 become deterministic and $\tilde{G} \vDash \Box(p \to \Diamond q)$. Therefore, by completeness of rule F-RESP of (Manna & Pnueli 1989), there exist a well-founded domain $\langle A, \preceq\rangle$, a ranking function $\delta : \Sigma \to A$, and m assertions $\tilde{\varphi}_1, \ldots, \tilde{\varphi}_m$ such that the following hold with respect to G:

$$\begin{aligned} &p \to q \vee \tilde{\varphi} \\ &\{\!\{ \tilde{\varphi}_i \wedge \delta = a \}\!\}\ \tilde{\mathcal{T}}\ \{\!\{ q \vee (\tilde{\varphi} \wedge \delta \prec a) \\ &\qquad\qquad \vee (\tilde{\varphi}_i \wedge \delta \preceq a) \}\!\} \\ &\{ \tilde{\varphi}_i \wedge \delta = a \}\ \tilde{\tau}_i\ \{ q \vee (\tilde{\varphi} \wedge \delta \prec a) \} \end{aligned}$$

If $\tilde{\tau}_i \in \tilde{\mathcal{J}}$,

$$\tilde{\varphi}_i \to q \vee En(\tilde{\tau}_i)$$

If $\tau_i \in \mathcal{C}$,

$$\tilde{G}(\tilde{\mathcal{C}} - \tilde{\tau}_i) \vDash \Box(\tilde{\varphi}_i \to \Diamond(q \vee En(\tau_i))),$$

where $\tilde{G}(\tilde{\mathcal{C}} - \tilde{\tau}_i)$ is the same as $\tilde{G}$, but with τ_i taken out of $\mathcal{C}$ and made unfair.

Now we can construct the objects we need for S-RESP-CONTROL. The domain $\langle A, \preceq\rangle$ and the ranking function δ stay the same. We define φ_i to be $\tilde{\varphi}_i \wedge win$. Let us verify that, with these choices, the premises of S-RESP-CONTROL hold in G. The first four control conditions are easy, since their structure maps closely that of the

verification conditions for F-RESP. For the last one, consider a compassionate τ_i and a G-reachable φ_i-state s, with $\delta(s) = a$. We prove that the same strategy f above allows Player 1 to force the game into a $q \vee (\varphi \wedge \delta \prec a) \vee (En(\tau_i) \wedge \varphi_i \wedge \delta \preceq a)$-state in G^i. Consider a continuation play σ, starting from s and compatible with f. Then σ is also a computation of $\tilde{G}(\tilde{\mathcal{C}} - \tilde{\tau}_i)$. If there is a q-state in σ, we are done. If there is none, then all states must be φ-states. If δ eventually assumes a value smaller than a, again we are done. Otherwise, all states in σ are φ_i-states. Hence σ must eventually reach a $(q \vee En(\tau_i))$-state s', and it must be the case that $s' = a$. □